フロアプランを考慮したマルチプレクサ削減 FPGA 高位合成手法

藤原 晃一 † 阿部 晋矢 † 川村 一志 † 柳澤 政生 † 戸川 望 †

† 早稲田大学大学院基幹理工学研究科情報理工・情報通信専攻

近年，画像処理や通信プロトコル処理などデータを高速処理する必要がある場面で，高位合成を利用した FPGA 設計が増加している．既存の FPGA 向け高位合成手法として，FPGA でのモジュールの配置（フロアプラン）を考慮した手法や，FPGA のマルチプレクサ（MUX）がボトルネックである特徴に着目し MUX を削減する手法がある．しかし，モジュールの配置と MUX の削減を同時に実現する手法は提案されていない．本稿では，FPGA 設計に HDR アーキテクチャを採用し，MUX を削減・制限する高位合成手法を提案する．提案手法では，レジスタ分散型アーキテクチャである HDR アーキテクチャを用いて，高位合成段階でモジュールの配置を考慮し，配線遅延を見積もる．また演算器バインディングでは MUX 数の削減を，レジスタバインディングでは MUX の入力数の制限を実現する．提案手法を計算機上に実装し，従来手法と比較した結果，スライス数を最大 38%，平均 17% 削減，遅延を最大 9%，平均 5% 削減を実現した．

A floorplan-driven FPGA high-level synthesis algorithm for multiplexer reduction

Koichi FUJIWARA† Shinya ABE† Kazushi KAWAMURA† Masao YANAGISAWA† Nozomu TOGAWA†

† Dept. of Computer Science and Communications Engineering, Waseda University

Recently, high-level synthesis (HLS) techniques for FPGA designs are required in reconfigurable network processing and image processing. Conventional HLS algorithms for FPGA designs realize either module floorplan-driven HLS or reducing multiplexer's cost but no HLS algorithm targeting FPGAs realizes both of them. In this paper, we propose a floorplan-driven high-level synthesis algorithm for multiplexer reduction. By utilizing a distirbuted-register architecture called HDR architecture, we can easily consider module floorplan in HLS. In order to reduce multiplexer's cost, we propose a novel binding method called datapath-oriented scheduling/FU binding and utilize datapath-oriented register binding. Experimental results demonstrate that our algorithm can realize FPGA designs which reduce the number of slices by up to 38% and circuit delay by up to 9% compared with the conventional approach.

1 はじめに

近年，証券取引などで利用されるアプリケーション，画像処理や通信プロトコル処理などの場面で FPGA の利用が増加している．証券取引などのアプリケーションは短期間で回路の修正・仕様変更が必要となるものがあり，FPGA を利用することで回路の修正・仕様変更が容易になる．また，画像処理や通信プロトコル処理の場面では大量のデータの高速処理が要求され，CPU によるソフトウェア処理ではなく，FPGA によるハードウェア処理を用いることで処理速度の要求に応えることができる．さらに市場規模が大きくないアプリケーションに対して FPGA を利用することで低コスト化を図れるなど，多くのメリットから FPGA の利用は大きく拡大している．

一方，近年では LSI 設計において設計規模の増大や設計コスト削減の問題が深刻化しており，これらに対する有効な手段の 1 つに高位合成がある．高位合成は抽象レベルの高い動作記述からハードウェア設計を可能にし，結果として設計の高速化や設計誤りの削減が実現される．高位合成を FPGA 設計に利用する必要性がますます増加している．

FPGA を対象とした高位合成では，以下の 2 つの問題点がある．

問題点 1 モジュールの配置（フロアプラン）の考慮の必要性

問題点 2 マルチプレクサ（MUX）のボトルネック

近年の FPGA は LSI プロセスの微細化が進み，入出力バッファや配線バッファが入っていてもゲート遅延に対する配線遅延の割合が増加している．高位合成段階でモジュールの配置を考慮することが強く求められる．一方で，FPGA では 3 章に後述する通り回路全体に占める MUX の面積・遅延割合が大きく，回路における MUX のコストが大きい．FPGA を対象とした高位合成では特に MUX の数・入力数を削減する必要がある．

既存の FPGA 向け高位合成手法として [3, 4, 6, 7] がある．[4, 6] では，FPGA でのモジュールの配置（フロアプラン）を考慮した高位合成手法を提案し，[3, 7] では，FPGA の MUX がボトルネックである特徴に着目し，MUX のコストを削減する高位合成手法を提案している．しかし，既存手法ではそれぞれが独自の目的関数を用いたバインディングにより，モジュールの配置の考慮・MUX 削減を行っているため，高位合成段階でモジュールの配置の考慮と MUX 削減を同時に達成する高位合成手法は実現できていない．

[1, 2] では，フロアプラン指向のアーキテクチャとして HDR (Huddle-based Distributed-Register) アーキテクチャを提案している．HDR アーキテクチャはレジスタ分散型アーキテクチャの 1 つで，回路全体をハドルと呼ぶ区画で分割し，ハドル毎にフロアプランを行い，配線遅延を見積もることができる．任意の矩形を取るハドルに対しフロアプランするため，小面積で遅延の少ないアーキテクチャとなる．これに対し，FPGA ではチップ全体に分散してレジスタが配置されており，HDR アーキテクチャは FPGA 設計に適している．しかし，[1, 2] の手法は一般の ASIC を対象とし

©2014 Information Processing Society of Japan

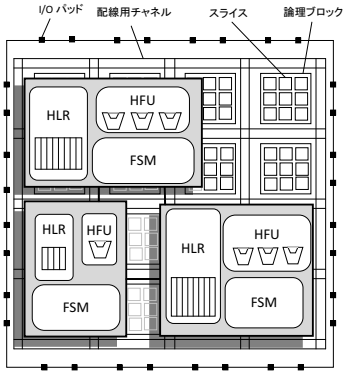


図 1: FPGA 上に構成された HDR アーキテクチャ.

ており，FPGA を対象とした HDR アーキテクチャの高位合成手法は提案されていない.

本稿では，HDR アーキテクチャを対象に，MUX を削減・制限する FPGA 向け高位合成手法を提案する．提案手法は，モジュールの配置・配線遅延と MUX のボトルネックを同時に考慮した FPGA 向け高位合成手法である．[5] で提案したパス考慮レジスタバインディングに加え，新たに各 FU（演算器）間のデータ転送を考慮する FU バインディング手法としてパス考慮スケジューリング/FU バインディングを提案する．パス考慮スケジューリング/FU バインディングでは，すでにデータパスがある 2 つの FU 同士を積極的に連続する 2 つの演算ノードに割り当てることで MUX の削減を図る．提案手法を計算機上に実装し，従来手法 [2] と比較した結果，スライス数を最大 38%，平均 17% 削減，遅延を最大 9%，平均 5% 削減した.

2 問題定義

本稿では，対象のアーキテクチャを HDR（Huddle-based Distributed-Register）アーキテクチャ [1, 2] とする．HDR アーキテクチャは，レジスタ分散型アーキテクチャの 1 つであり，回路全体をハドル（Huddle）と呼ぶ区画で分割し，ハドル毎にフロアプランを行い，配線遅延を正確に見積もることができる．ハドルは配線遅延の影響の無い範囲で任意の矩形を取り，FU やレジスタ，コントローラを共有する．HDR アーキテクチャの構成を図 1 に示す．ハドルは以下の要素で構成される.

Huddled Local Register（HLR） 各ハドル専用のローカルレジスタと MUX の集合である.

Huddled Functional Unit（HFU） ハドルに集められた FU の集合である．ハドル内で処理する演算に必要な FU を必要数持ち，同一のハドル内の HLR のみにアクセスできる.

Finite State Machine（FSM） 各ハドル専用のコントローラである．同一ハドルの HFU と HLR を制御する.

FPGA は論理ブロック, I/O パッド, 配線チャネルで構成される．各論理ブロックは複数のスライスで構成され，各スライスは LUT とフリップフロップで構成される．FPGA 向け HDR アーキテクチャでは，FPGA 上で各ハドルがスライス単位で構成される．HFU は FU と MUX の集合であり，スライスの LUT で構成される．HLR はレジスタと MUX の集合であり，スライスの LUT とフリップフロップで構成される．FSM はコントローラであり，スライスの LUT とフリップフロップで構成される．各ハドル間は配線チャネルを用いて接続され，外部とのデータの入出力は I/O パッドを介して行われる.

表 1: FPGA・ASIC の各回路構成要素の遅延割合と面積割合.

FU	FPGA 遅延割合	FPGA LUT 数割合	ASIC 遅延割合	ASIC 面積割合
加算器	1.00	1.00	1.00	1.00
減算器	1.00	1.00	1.01	1.21
乗算器	2.84	11.1	2.00	7.15
除算器	19.6	24.1	10.5	9.06
右シフト器	1.47	2.44	0.42	1.02
比較器	1.06	0.38	0.15	0.40
AND	0.55	1.00	0.02	0.24
2 入力 MUX	0.66	1.00	0.03	0.39

HDR アーキテクチャを対象とした FPGA 高位合成問題を以下のように定義する.

定義 1 HDR アーキテクチャを対象とした FPGA 高位合成問題とは，コントロールデータフローグラフ（以下 CDFG），クロック周期制約，ステップ制約，演算器制約が与えられた時，回路の面積（スライス数）と遅延を最小化するように CDFG をスケジューリングおよびバインディングし，各 FU をハドルに割り当て，レジスタ・コントローラを合成することである． □

3 動機付け

前章で示した問題を解くために，FPGA 上に実現される回路において面積・遅延のボトルネックとなる構成要素を特定する．表 1 に FPGA と ASIC それぞれの加算器の値（遅延・LUT 数・面積）を 1 として，各回路構成要素の相対値を算出した結果を示す．各回路構成要素の入出力は全て 16 ビットとした．実験環境は FPGA に関して，Xilinx 社 ISE Design Suite 14.2 の XST で論理合成した．FPGA ボードは Virtex-6 を想定した．ASIC に関して，Synopsys 社の Design Compiler D-2010.03-SP5 のトポグラフィカルモードを用いて, クロック制約は 5.0ns, ライブラリは STARC 90nm テクノロジを用いて論理合成した.

表 1 の結果から MUX に関して，FPGA では ASIC に比べ面積は約 2.6 倍，遅延は約 22 倍の相対値を示した．FU に関して，FPGA では ASIC に比べて平均で面積は約 1.83 倍，遅延は約 2.3 倍の相対値を示した．つまり，FPGA の方が ASIC に比べて回路全体に占める MUX のコストが大きくボトルネックである．FPGA では MUX 数を削減する高位合成が必要である.

次に, MUX の入力数に対する遅延・LUT 数の測定結果を表 2 に示す．実験環境は Xilinx 社の ISE Design Suite 14.2 において XST で論理合成した．入出力は全て 16 ビットとした．表 2 から，2–4 入力の MUX は 16 個の LUT を必要とし，5 入力以上の MUX は 32 個以上の LUT を必要とする．つまり，2–4 入力の MUX は面積が等しく，5 入力以上の MUX の半分以下であることがわかる．2–4 入力の MUX の面積が等しいことは，Virtex-6 が 6 入力 LUT を搭載しているためである．FPGA では 5 入力の MUX の数を削減し，でき

©2014 Information Processing Society of Japan

表 2: FPGA における入力数ごとの MUX の値.

MUX	遅延 (ns)	LUT 数
2 入力 MUX	1.002	16
3 入力 MUX	1.184	16
4 入力 MUX	1.186	16
5 入力 MUX	1.575	32
6 入力 MUX	1.650	32
7 入力 MUX	1.717	47
8 入力 MUX	1.443	33
16 入力 MUX	1.611	66

る限り 4 入力の MUX を作ることが必要である.

以上から，FPGA において MUX のコストを考慮する高位合成手法において以下のアプローチが有効である.

1. MUX の数を削減する.
2. MUX を 4 入力以下にする.

4 提案アルゴリズム

4.1 全体フロー

[2] は HDR アーキテクチャを対象とした高位合成手法 MH^4 を提案した．MH^4 は，入力として与える各 FU に対しそれぞれハドルを構成し，全てのハドルが重なった状態を初期解とする．このときハドルは 1 つの FU だけを持ち，レジスタやコントローラは持たないとする．初期解をもとにスケジューリング，バインディング，フロアプラン指向ハドル合成を繰り返し，解を改良する．データ転送制約違反のない解を得た場合，最終的な解を出力する．しかし，MH^4 は MUX を大きなコストと考えておらず，バインディングの際に各ハドル毎に FU やレジスタをできる限り共有するようバインディングする．結果として，MUX が増加することが予想される.

そこで HDR アーキテクチャを対象とする FPGA 高位合成として，MUX を大きなコストと考え，各 FU（演算器）間のデータ転送を考慮する FU バインディング手法としてパス考慮スケジューリング/FU バインディングを提案する．パス考慮スケジューリング/FU バインディングでは，すでにデータパスがある 2 つの FU 同士を積極的に連続する 2 つの演算ノードに割り当てることで MUX の削減を図る．また，ハドル間のデータ転送遅延を計算する際に FPGA のスライス単位で距離を扱うことで，より FPGA に適した配線遅延の見積もりを行う．レジスタバインディングでは，[5] で提案したパス考慮レジスタバインディングを用いてレジスタに付加する MUX のコスト削減を図る．パス考慮レジスタバインディングは，前章で述べた FPGA において 4 入力以下において MUX のコストが小さくなる点に着目し，全ハドルのレジスタに付加する MUX の入力数を全て 4 入力以下に制限することで回路全体の MUX のコストの削減を図るバインディング手法である.

図 2 に HDR アーキテクチャを対象とした MUX 入力数制限高位合成アルゴリズムを示す．提案アルゴリズムでは，入力として CDFG，クロック周期制約，ステップ制約，演算器制約を与え，RTL 回路の記述とハドルの構成・配置情報を出力する．まず，ハドルフロアプランでは各ハドルに演算器が 1 つ割り当てられた状態でフロアプランをする (フェーズ 1)．このハドル配置を基に，スケジューリング/FU バインディング (フェーズ 2)，レジスタバインディング (フェーズ 3)，コントローラ合成 (フェーズ 4)，フロアプラン指向ハドル合成 (フェーズ 5) を繰り返し，解を改良する．データ転送制約を満たす回路を解として得た場合，実際の面積に調整して結果を出力する (フェーズ 6).

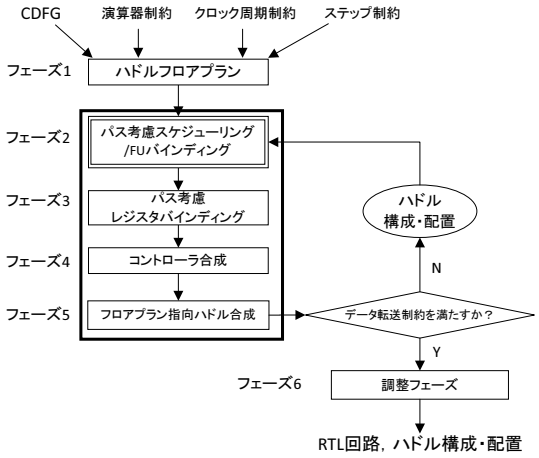


図 2: HDR アーキテクチャを対象とした MUX 削減高位合成アルゴリズム.

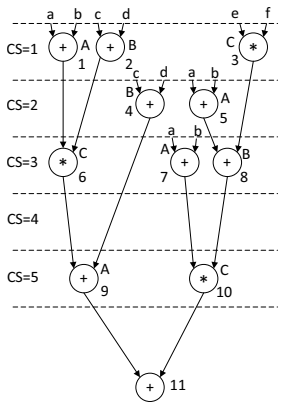
提案アルゴリズムにおいて，回路の MUX のコストは FU バインディング (フェーズ 2) とレジスタバインディング (フェーズ 3) 結果に起因する．そこでフェーズ 2 では各 FU（演算器）間のデータ転送を考慮する FU バインディングを行いて MUX 数の削減を図り，フェーズ 3 では [5] のパス考慮レジスタバインディングを用いて全ハドルの各レジスタに付加する MUX を 4 入力以下に制限し，回路全体の MUX のコストを削減する．また，フェーズ 2 では，前のイタレーションでのハドル構成・配置を基にスライス単位で距離を扱いハドル間のデータ転送遅延を見積もる．このデータ転送遅延を考慮してスケジューリング/FU バインディングすることで，最終的な回路の配線遅延を考慮したスケジューリング/FU バインディングを実現している.

フェーズ 1，4，5，6 は [2] と同様である．前述の通りフェーズ 3 は [5] で提案した手法を用いる．本章では，パス考慮スケジューリング/FU バインディング (フェーズ 2) を提案する.

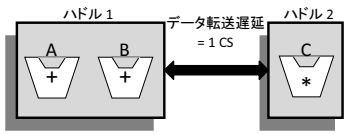
4.2 パス考慮スケジューリング/FU バインディング

今，図 3 に示すようなスケジューリング/FU バインディングを考える．レジスタとコントローラは提案アルゴリズムの後のフェーズで合成されるため，ここでは考えない．図 3(a) は，入力となる CDFG を表す．図 3(b) は，スケジューリング/FU バインディングの入力となるハドルの構成・配置情報を表す．FU A, B はハドル 1，FU C はハドル 2 にフロアプランされているとする．ハドル間のデータ転送には 1 CS かかるとする．なお，このハドルの構成・配置情報は 1 つ前のイタレーションで決定したもので，このイタレーションではこの情報をもとに処理を進める．図 3(a) のようにコントロールステップ $CS = 5$ までスケジューリング・FU バインディングが完了し，次にノード 11 のスケジューリング/FU バインディングを考える.

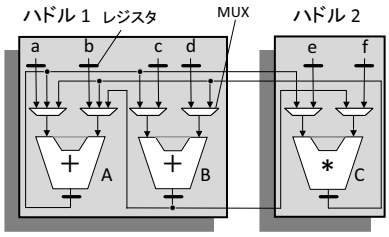
ノード 11 のスケジューリングと FU バインディン



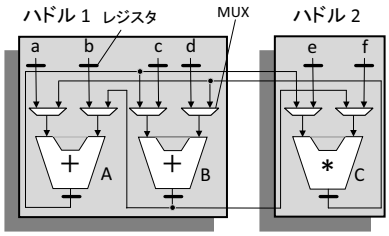
(a) 入力の CDFG.



(b) ハドルの構成・配置情報.



(c) ノード 11 に FU A を割り当てた場合.



(d) ノード 11 に FU B を割り当てた場合.

図 3: スケジューリング/FU バインディング例.

グを考える．ノード 11 は加算ノードであるので，FU A–C の内，加算器の FU A と B がノード 11 に割り当てられる FU の候補になる．ここで，FU A と B は共にハドル 1 に配置されおり，ハドル 2 に配置されている FU C からのデータ転送に 1 CS かかる．よって，ノード 11 を FU A と B どちらに割り当てたとしても，ノード 11 を $CS = 6$ には割り当てられず，ノード 11 は $CS = 7$ に割り当てられる．図 3(c) は，ノード 11 に FU A を割り当てた時のデータパスを表す．ここではレジスタについて議論しないため，各データ転送に 1 つレジスタを設けている．図 3(d) は，ノード 11 に FU B を割り当てた時のデータパスを表す．

図 3(a) で，ノード 5，ノード 3，ノード 8 に注目する．これらのノードはすでにスケジューリング，FU バインディングが完了しており，ノード 5 は FU A に，ノード 3 は FU C に，ノード 8 は FU B に割り当てられている．そのため，FU A から FU B の左入力に至るデータパス，FU C から FU B の右入力に至るデータパスがすでに存在する．ノード 11 の FU バインディングにおいて FU B をノード 11 に割り当てると，すでに合成済みのデータパスを再利用できるため，図 3(d) のように FU B に付加する MUX の入力数が増えない．結果として図 3(d) では，図 3(c) に比べて MUX の入力数を削減している．FU バインディングではハドルによるデータ転送遅延を考慮した上で，合成済みのデータパスを再利用することにより，MUX の入力数を削減できることがわかる．

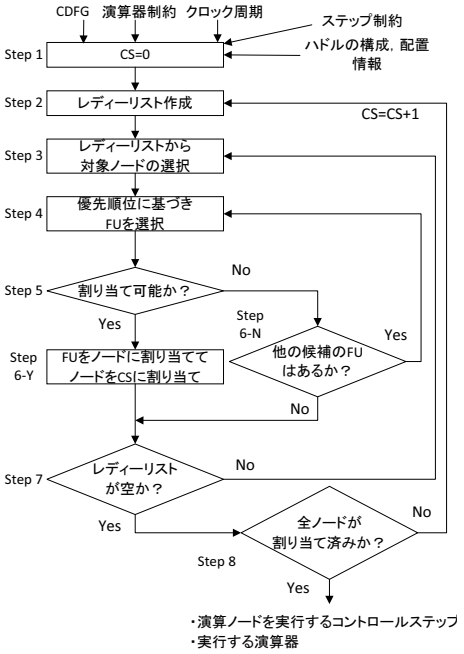


図 4: パス考慮スケジューリング/FU バインディング.

図 4 にパス考慮スケジューリング/FU バインディングのアルゴリズムを提案する．提案手法は，ハドルによるデータ転送遅延を考慮した上で，合成済みのデータパスを再利用することにより MUX 数の削減を図る．提案手法では，入力として CDFG，演算器制約，クロック周期制約，ハドルの構成，配置情報をもとにデータ転送遅延情報を与えて，演算ノードを実行するコントロールステップ，実行する FU を出力する．ここで，ハドルの構成・配置情報は 1 つ前のイタレーションで得られたものであり，パス考慮スケジューリング/FU バインディングでは，各 FU が配置されるハドル・ハドルの位置はこれらを流用する．各ハドル内のレジスタとコントローラは後のフェーズで再合成するため，パス考慮スケジューリング/FU バインディングでは考えない．

提案手法では，スケジューリングと FU バインディングを同時に実行する．各ノードの FU を決定する時に，出力先のノードが割り当てられる FU は未決定のため，親ノードに割り当てられた FU のみを考慮して，FU バインディングを行う．提案手法では，FU を選択する際に，親ノードに割り当てられた FU からデータパスがある，かつ最も早い CS に割り当てられる FU を優先的に選択する．親ノードに割り当てられた FU からデータパスがある，かつ最も早い CS に割り当てられる FU が無い場合は，割り当てた際の最終的な全体の CS 数の見積もりが最小の FU を選択する．

図 4 にパス考慮スケジューリング/FU バインディングアルゴリズムを示し，以下で各ステップを提案する．

Step 1（CS の初期化） $CS = 0$ よりパス考慮スケジューリング/FU バインディングを実行する．

Step 2（レディーリスト作成）

$CS = n$ におけるノードのスケジューリングを考える．$CS = n - 1$ までは，リストスケジューリングベースで $CS = n - 1$ までに割り当て可能なノードはスケジューリングされ，スケジューリングされたノード全てにいずれかの

FU が割り当てられていると仮定する．データ依存関係を違反しない範囲で，$CS = n$ に割り当てられる可能性のあるノードのリストであるレディーリストを作成する．

Step 3（対象ノードの選択）

レディーリストから優先度 $priority(n_i)$ に従ってノード n_i を選択する．ノード n_i における優先度 $priority(n_i)$ は各ノード n_i において，終端ノードまでのクリティカルパス長を算出し，クリティカルパス長が長いノードに高い優先度を与える．

$$cp(n_i, fj) = c_j + \max_{n_k \in succ(n_i)} \{\min_{f_l \in F}(id_{f_i,f_l} + cp(n_k, f_l)\} \tag{1}$$

$$priority(n_i) = \min_{k \in F} cp(n_i, f_k) \tag{2}$$

F は入力で与えられた FU の集合を表す．$cp(n_i, f_j)$ はノード n_i に FU f_j を割り当てた場合の入力の CDFG の終端ノードからノード n_i までのクリティカルパス長を計算したものである．c_j は FU f_j の演算サイクル数，$succ(n_i)$ はノード n_i のすべての子孫ノード，id_{f_j,f_l} は FU f_j から FU f_l へのデータ転送に必要なステップ数を表す．

Step 4–7（FU の割り当て処理）

選択されたノード n_i に対し，割り当てる FU を決定する．入力の CDFG では，各演算ノードの親ノードの数は 2 個以下である．従って，親ノードが 2 個，1 個，0 個の場合を考える．

親ノードが 2 個の場合を考える．ノード n_i の親ノードを n_j と n_k とする．ノード n_j には FU f_j，ノード n_k には FU f_k が割り当てられているとする．FU f_j と f_k からすでにデータパスがある，かつノード n_i の演算に対応する FU の集合を F^1_{path} とする．FU $f_l \in F^1_{path}$ をノード n_i に割り当てて $CS = n$ にスケジューリングする場合，FU f_j と f_k から FU f_l へのデータ転送遅延を満たすか調べる．FU f_j と f_k から FU f_l へのデータ転送遅延を満たす場合，ノード n_i において FU f_l は，新たな MUX を付加せず最も早い CS にスケジューリングできる FU である．FU f_j と f_k から FU f_l へのデータ転送遅延を満たす場合は，FU f_l をノード n_i に割り当て，ノード n_i を $CS = n$ にスケジューリングする．FU f_j と f_k から FU f_l へのデータ転送遅延を満たさない場合は，F^1_{path} の他の FU を同様に調べる．

F^1_{path} の FU 全てが $CS = n$ においてノード n_i に割り当て不可能な場合，FU f_j あるいは f_k のいずれかからすでにデータパスがある，かつノード n_i の演算の種類に対応する FU の集合を F^2_{path} とする．FU $f_m \in F^2_{path}$ をノード n_i に割り当てて $CS = n$ にスケジューリングする場合，FU f_j と f_k から FU f_l へのデータ転送遅延を満たすか調べる．FU f_j と f_k から FU f_m へのデータ転送遅延を満たす場合，ノード n_i において FU f_m は，右または左の入力に新たな MUX を付加せず最も早い CS にスケジューリングできる FU である．FU f_j と f_k から FU f_m へのデータ転送遅延を満たす場合は，FU f_m をノード n_i に割り当て，ノード n_i を $CS = n$ にスケジューリングする．FU f_j と f_k から FU f_m へのデータ転送遅延を満たさない場合は，F^2_{path} の他の FU を同様に調べる．

F^2_{path} の FU 全てが $CS = n$ においてノード n_i に割り当て不可能な場合，ノード n_i の演算の種類に対応する，かつ見積もりレイテンシが最小となる FU f_n を選択する．見積もりレイテンシとは，あるノード n_i に FU f_n を割り当てた際の最終的に必要となる CS 数の見積もり値である．ノード n_i における FU f_n のレイテンシ見積もり $lat(n_i, f_n)$ を以下のように定める．

$$lat(n_i, f_n) = cstep(n_i, f_n) + cp(n_i, f_n) \tag{3}$$

$cstep(n_i, f_n)$ はノード n_i に FU f_n が割り当て可能になるコントロールステップを表す．見積もりレイテンシにより選択された FU f_n をノード n_i に割り当てて $CS = n$ にスケジューリングした場合，入力先の FU f_j と f_k から FU f_n へのデータ転送遅延を満たすか調べる．FU f_j と f_k から FU f_n へのデータ転送遅延を満たす場合，FU f_n をノード n_i に割り当て，ノード n_i を $CS = n$ にスケジューリングする．

FU f_j と f_k から FU f_n へのデータ転送遅延を満たさない場合，ノード n_i は $CS = n$ にスケジューリングできないと判断し，レディーリストの次のノードを選択する．

親ノードが 1 個と 0 個の場合も，親ノードが 2 個の場合と同様である．

Step 8（終了判定）

レディーリストが空になった場合，$CS = n$ にスケジューリング可能なノード全てをスケジューリングしたと判断し，$CS = n + 1$ として同様の処理を行う．全てのノードをスケジューリングし終えたら，スケジューリング結果を出力する．

5 計算機実験

HDR アーキテクチャを対象とした FPGA 設計フローを図 5(a) に示す．まず提案手法によって RTL 記述を出力する．次に，Xilinx 社 ISE Design Suite 14.2 の XST で論理合成を行い，ゲートレベルの回路情報を得る．FPGA ボードは Xilinx 社の Virtex-6 を想定する．なお，本実験では dsp ブロックの使用を禁止とした．次に，ISE 内の PlanAhead で配置・配線を行い，最終的な回路情報を得る．配置配線の際，クロック周期を最小とする合成結果を得ている．最後に，ISE により BIT ファイル化し，FPGA 上の HDR アーキテクチャが実現する．

SR，従来手法 [2]，提案手法を各アプリケーション用いて比較した．ここで SR とはハドルを用いずレジスタ集中型アーキテクチャに対して従来手法 [2] を適用したものである．レジスタ集中型アーキテクチャとは，ハドルを構成せず各 FU でできるだけレジスタを共有させたアーキテクチャである．SR を用いた FPGA 設計フローを図 5(b) に示す．各手法の実行に使用した計算機環境は，AMD Opteron 2360SE 2.5GHz×2，メモリが 16GB である．論理合成，配置・配線に使用した計算機環境は，Intel Core i5-2520M 2.5GHz×2，メモリが 4GB である．

実験結果を表 3 に示す．提案手法を用いた場合，従来手法 [2] と比較して，スライス数を最大 38%，平均 17% 削減，遅延を最大 9%，平均 5% 削減した．また，レジスタ集中型の回路と比較して，スライス数を最大 53%，平均 39% 削減，遅延を最大 26%，平均 7% 削減した．パス考慮スケジューリング/FU バインディングを用いていない [5] と比較して，スライス数を最大 11%，平均 2% 削減，遅延を最大 12%，平均 3% 削減した．

6 おわりに

本稿では，FPGA のモジュールの配置と MUX のボトルネックを同時に考慮した FPGA 向け高位合成手法として，フロアプランを考慮したマルチプレクサ削減 FPGA 高位合成手法を提案した．計算機実験の結果から，提案手法は従来手法 [2] と比較して，スライス数を最大 38%，平均 17% 削減，遅延を最大 9%，平均 5% 削減できた．今後の課題として，より MUX の総面積・遅延を削減するバインディング手法の提案が挙げられる．

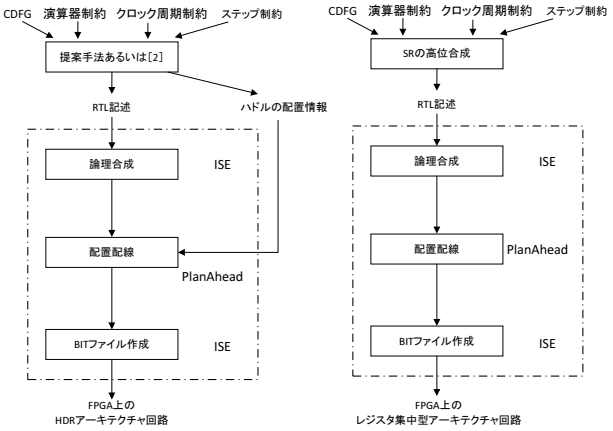
謝辞

本研究は独立行政法人新エネルギー・産業技術総合開発機構 (NEDO) の先導的産業技術創出事業の支援を受けて行われた．

©2014 Information Processing Society of Japan

表 3: 計算機実験結果.

App.	#Nodes	FUs	Clock period constraints [ns]	Steps tconstraint	Algorithm	Delay [ns]	#Slices	#LUTs	#REGs	Total synthesis time [sec]
hal	11	Add×1 Sub×1 Mul×2 Comp×1	6	7	SR [2] [5] 提案手法	4.996 4.856 5.153 4.944	179 124 108 107	431 365 356 340	146 159 152 142	131 126 164 135
parker	22	Add×2 Sub×2 Comp×1	3	7	SR [2] [5] 提案手法	2.170 2.434 2.170 2.264	122 69 62 62	235 246 225 225	256 288 271 270	136 254 158 155
dct	48	Add×4 Mul×4	8	12	SR [2] [5] 提案手法	5.670 5.947 6.164 5.468	534 369 361 321	1269 1187 1169 1101	355 503 513 495	171 207 243 186
jacobi	52	Add×2 Sub×1 Mul×2 Div×2	32	30	SR [2] [5] 提案手法	21.72 23.37 21.27 22.61	529 373 356 340	1312 1289 1248 1249	296 336 336 334	167 188 183 244
fir	75	Add×5 Mul×4	7	31	SR [2] [5] 提案手法	6.085 5.769 5.344 5.267	556 420 234 261	1440 1435 803 913	309 398 374 432	163 210 199 213
ewf3	102	Add×4 Mul×4	7	54	SR [2] [5] 提案手法	6.239 5.122 5.253 4.643	590 428 415 379	1643 1501 1301 1188	376 436 504 471	186 279 219 239
copy	378	Add×5, Sub×1 Mul×5, Comp×1 Rshift×2 AND×1	12.5	172	SR [2] [5] 提案手法	7.967 6.992 7.394 6.858	1690 1810 1421 1404	4932 5185 4069 3881	1559 2241 2698 2689	310 898 547 557



(a) HDR アーキテクチャ設計フロー.　(b) SR 設計フロー.

図 5: FPGA 設計フロー.

本研究を進めるにあたり貴重な議論をいただいた日本電気 (株) グリーンプラットフォーム研究所・竹中崇博士ならびに同研究所のみなさまに感謝いたします.

参考文献

[1] S. Abe, M. Yanagisawa, and N. Togawa, "An energy-efficient high-level synthesis algorithm for huddle-based distributed-register," in *Proc. of 2012 IEEE International Symposium on Circuits and Systems*, pp. 576–579, 2012.

[2] S. Abe, Y. Shi, M. Yanagisawa, and N. Togawa, "MH^4:multiple-supply-voltages aware high-level synthesis for high- integrated and high-frequency circuits for HDR architectures," *IEICE Electronics Express*, vol. 9, no. 17, pp. 1414–1422, 2012.

[3] D. Chen, J. Cong, and Y. Fan, "Low-power high-Level synthesis for FPGA architecture," in *Proc. of International Symposium on Low Power Electronics and Design*, pp. 134–139, 2003.

[4] J. Cong, B. Liu, G. Luo, and R. Prabhakar, "Towards layout-friendly high-level synthesis," in *Proc. of International Symposium on Physical Design*, pp. 165–172, 2012.

[5] 藤原晃一, 阿部晋矢, 川村一志, 柳澤政生, 戸川望, "フロアプランを考慮したマルチプレクサ入力数制限 FPGA 向け高位合成手法," 信学技報, VLD2014-50, 2014.

[6] M. Xu and F. J. Kurdahi, "Layout-driven high level synthesis for FPGA based architecture," in *Proc. of Design, Automation & Test in Europe Conference & Exhibition*, pp. 446–450, 1998.

[7] L. Zhong and N. K. Jha, "Interconnect-aware high-level synthesis for low power," in *Proc. of International Conference on Computer-Aided Design*, pp. 110–117, 2002.

©2014 Information Processing Society of Japan